NBSIR 87-3638

Some Performance Comparisons for a Fluid Dynamics Code

Daniel W. Lozier
Ronald G. Rehm

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
National Engineering Laboratory
Center for Applied Mathematics
Gaithersburg, MD 20899

September 1987

U.S. DEPARTMENT OF COMMERCE
NATIONAL BUREAU OF STANDARDS

NBSIR 87-3638

SOME PERFORMANCE COMPARISONS FOR A FLUID DYNAMICS CODE

Daniel W. Lozier
Ronald G. Rehm

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
National Engineering Laboratory
Center for Applied Mathematics
Gaithersburg, MD 20899

September 1987

U.S. DEPARTMENT OF COMMERCE, Clarence J. Brown, *Acting Secretary*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

2. BENCHMARK PROBLEM

In this section we describe briefly the source of the benchmark problem, the major logical structure of the Fortran program, and the parameters of three different specific instances of the benchmark problem that vary widely in the amount of time and memory required for execution.

Research Background

As stated in the introduction, our purpose in benchmarking computers is solely in the interest of furthering our investigations into fundamental problems of fire science. Over a decade ago, stimulated by federal recognition of very large losses of life and property by fires each year throughout the nation, NBS became actively involved in a national effort to reduce such losses. The work at NBS ranges from very practical to quite theoretical; our approach, which proceeds directly from basic principles, is at the theoretical end of this spectrum.

Early work was concentrated on developing a mathematical model of convection arising from a prescribed source of heat in an enclosure, e.g. an open fire in a room. A set of partial differential equations [11] was derived from the basic physical principles of the conservation of mass, momentum and energy to describe the transport of smoke and hot gases. The model is valid under assumptions appropriate to very non-adiabatic, non-dissipative, buoyant flows of a perfect gas such as are generated when heat is added slowly. The equations contain all the terms of the more familiar Boussinesq equations, to which a detailed comparison is made in [11].

The equations were rewritten as hyperbolic evolution equations for density and velocity plus an elliptic equation for the pressure. From these, a finite-difference scheme was constructed for both 2D and 3D, the former of which is presented in [3] together with sample calculations. The scheme is validated [4, 12] by comparing special 2D and 3D test cases against analytic solutions that were developed specifically for this purpose, and the overall model has been validated by comparison with salt-water/fresh-water experiments of buoyancy-induced flows [1, 2].

Recent work has turned to the problem of computing the heat source as the flow evolves, rather than simply describing it a priori by means of a mathematical function [5,6]. The length and time scales of combustion processes are so short compared to the scale of convection in an enclosure that it is impossible to compute the heat source by refining the finite-difference mesh. Our approach is to model the combustion process as a separate small-scale problem [5,6]. Toward this end we are calculating the flame interface, the species consumption rates, and the heat release rate between two reacting species in a stretched vortex flow field. Also, as a special two-dimensional case [13] we have determined these quantities from an exact global analytical solution to a problem originally posed and analyzed by local methods by Marble [10] ; a more general problem has been analyzed also by local methods by Karagozian and Marble [9].

We have been computing Lagrangian particle trajectories in BF3D and displaying them as a way of visualizing the 3D fluid flow; the trajectory calculations are systems of ordinary differential equations. Until now these particles have been serving passively only to mark the progress of the flow but they are in fact a mechanism for distributing heat from burning fluid parcels into the flow. The additional computations needed for this project will increase significantly the computational demands of BF3D.

Fortran Program

BF3D consists of a main program and 40 subroutines for a total of approximately 3500 lines, excluding comments. It is written in standard Fortran 77 except for a single, isolated call to a system-dependent timing subroutine. The timing subroutine is used to generate a simple profile of time utilization among major components of the program. Some of the timing data presented in the Appendix was obtained by examining these profiles.

A parallelepiped in (x,y,z) space is divided into a uniform mesh that is fixed by input parameters. Density, the x, y and z components of velocity, and two flux variables are stored at every meshpoint for three consecutive timepoints. In addition to 18 three-dimensional arrays for these quantities, BF3D has 3 three-dimensional arrays for the linear elliptic pressure solver and 9 two-dimensional arrays for boundary conditions. The Lagrangian particle-tracking algorithm requires 7 memory locations for each

particle. These are the main sources of memory utilization in the program.

Figure 1 illustrates the logical structure of the program. After reading the input data and initializing the Fortran arrays to account for the boundary and initial conditions, BF3D starts the computation by computing the pressure at the first nonzero timepoint. The finite-difference scheme achieves second-order accuracy in space and time by using an appropriately staggered grid. Thus density and velocity are updated using their values from two timesteps back and pressure from one timestep back. This is shown at the beginning of the loop in Figure 1. Since the mesh is never refined, numerical stability is maintained by reducing the timestep in accordance with a CFL type of stability criterion. If the criterion is violated, the timestep is reduced and the computation is restarted. Otherwise the pressure at the new timestep is computed, the particle trajectories are updated, the output files are written, and the cycle is complete.

```
        Read input data.
        Initialize variables and data structures.
        Set n = 1.
Start:  Compute pressure at time t = nh,
          where h is the current timestep.
 Loop:  Compute density at t = (n+1)h.
        Compute velocity at t = (n+1)h.
        Check CFL stability criterion;
          if satisfied then continue,
          else reduce h and go to Start.
        Compute pressure at t = (n+1)h.
        Update particle trajectories.
        Write data to output files.
        Increment n.
        Go to Loop.
```

Figure 1. Logical structure of BF3D.

Benchmark Series

Benchmark runs of three different sizes make up the benchmark series.

The **small benchmark** models a fire in a room with a square plan form and a low ceiling using a 20x20x8 spatial mesh, 125 timesteps, and 1000 particles. These parameters are not representative of the size or time needed for a physically meaningful calculation. We include it because it is an easy first test for computers of all sizes. It uses about 1 megabyte of memory for 64-bit arithmetic and takes approximately 15 seconds of supercomputer processor time.

The **large benchmark** uses a 40x40x40 spatial mesh, 466 timesteps, and 12000 particles to model a fire in a cubical room. About 12 megabytes of memory and 10 minutes of processor time are needed to run this benchmark in 64-bit arithmetic on a supercomputer. It is the most representative of the three benchmarks in our series of actual runs we have made.

The **giant benchmark** is also for a cubical room. It uses a 48x48x48 spatial mesh, 50 timesteps, and 10000 particles. About 3 minutes of processor time and 20 megabytes of memory are needed for a 64-bit run on a supercomputer. The spatial mesh is typical of runs we will be making in the near future, but they will run for many more timesteps.

3. COMPUTERS BENCHMARKED

We ran the benchmark series on the following computers:

Alliant FX/8
Alliant Computer Systems Corporation
Littleton, Massachusetts

Convex C1 XP
Convex Computer Systems Corporation
Richardson, Texas

Cyber 205
Control Data Corporation
Minneapolis, Minnesota

SCS-40
Scientific Computer Systems Corporation
San Diego, California

Sun 3/160
Sun Microsystems
Mountain View, California

Vax 11/785
Digital Equipment Corporation
Maynard, Massachusetts

Distinguishing characteristics of these different systems will be briefly described.

Non-Scalar Computers

The Alliant, Convex, Cyber and SCS machines are vector computers. This means that the processor can execute vector (as well as scalar) instructions. A vector instruction applies a single operation, such as floating-point add, to a sequence of operands. In contrast, a scalar instruction applies a single operation to a single operand. A vector processor is a scalar processor with an added component called a pipeline. Each vector processor implements pipelining in a different way, but in each case the effect is to enable execution of a vector instruction in less time than the same processor could execute an equivalent sequence of scalar instructions. The speedup is achieved by a kind of micro-parallelism in which different stages of the pipeline work simultaneously on different data. In general, performance increases with vector length up to a limit that depends on the implementation of the pipeline.

The pipeline can be used effectively only when data are supplied to it at a sufficiently high rate. Generally this means the data must be arranged consecutively in memory. If it is not so arranged, the data may have to be moved about in memory to bring it into an arrangement that is acceptable for vector arithmetic. If this rearrangement can be achieved "at vector speed", i.e. by the pipeline under the control of suitable vector instructions, then not much performance is lost. If not, a "scalar bottleneck" prevents effective use of the pipeline.

A program may be vectorized (written to take advantage of vector instructions) either manually by a programmer or automatically by a compiler. Sometimes both approaches are used together to achieve an optimal speedup. The most important program construct that is a candidate for automatic vectorization is the innermost loop in a nest of (one or more) DO loops. To achieve a fair comparison of computer systems, we have avoided using any programming technique that does not meet the Fortran 77 standard. Consequently, the vectorizing performance of the compiler is critically important for the results of our benchmark series, and we give in the next major section an analysis of inner-loop vectorization for each vector computer we tested.

The Alliant differs from the Convex, Cyber and SCS computers in that it integrates multiple processors whereas the others are unit processors. It can have up to 8 tightly-coupled high-performance processors, and it provides compiler support for parallelizing the program so that the processors can operate in parallel on a single job. That is, in addition to producing vector instructions for each processor, the compiler analyzes the code to identify sections that are logically independent and therefore executable in parallel. Such sections typically are DO-loops or DO-loop nests. Single DO-loops are subdivided and distributed to all available processors, which run in vector mode if the loop is vectorizable. In the case of a DO-loop nest for which the innermost loop is vectorizable, the next outer loop is subdivided and distributed to all available processors. Of necessity, the Alliant compiler is able to create, synchronize and delete independent processes. These and other task-scheduling functions are available for manually parallelizing a Fortran program; we did not use them because the programming techniques are

nonstandard.

Scalar Computers

The Sun and Vax machines are scalar computers. We ran the benchmark series on these computers not to imply that they are suitable for calculations of the size and complexity of a program like BF3D but rather to afford an indication of the kind of speedups that can be expected from vector computers. In our benchmark tests all the vector computers were run in 64-bit precision and all the scalar computers were run in 32-bit precision. To run the scalar computers in 64-bit precision would require re-declaring all REAL variables to be DOUBLE PRECISION. This would extend quite considerably the time required to run the benchmarks. The real value of the Sun and Vax computers, in our opinion, is for program development, debugging and auxiliary operations such as graphics.

4. COMPILER COMPARISON

In this section we measure the vectorizing performance of the Alliant, Convex, Cyber and SCS Fortran compilers by determining the percentage of inner loops vectorized; see Table 1. The determination was made by examining the output listings produced by the compilers. We present the measure separately for four functional subdivisions of BF3D: the *evolution code* that advances the finite-difference solution from one timestep to the next; the *pressure code* that solves the elliptic equation at a given timestep; the *particle code* that advances the trajectories of the Lagrangian particles; and the *output code* that writes the output files. The number of loops in each subdivision is given in parentheses.

	Cyber	SCS	Alliant	Convex
Evolution (66)	50%	98%	100%	100%
Pressure (63)	27%	65%	87%	90%
Particle (10)	20%	50%	70%	60%
Output (3)	67%	67%	67%	67%
Overall (141)	38%	80%	92%	93%

Table 1. Percentage of Inner Loops Vectorized.

The results reported here are for versions of the compilers that were available toward the end of 1986. Specifically, we used the Alliant FX/Fortran Compiler V2.0.18 on 19 December 1986, the Convex Fortran Compiler V2.1 on 4 November 1986, and the Cyber Fortran 200 Compiler (Cycle 661C) on 18 January 1987. The SCS compilation was done on 31 March 1987 using the Cray Fortran Compiler V1.13. Because the SCS-40 has a machine instruction set that is identical to the Cray X-MP, the compiler needed no alterations.

Earlier versions of the Cyber compiler had a restriction that prevented vectorization of inner DO loops unless the upper and lower limits of the DO loop were fixed at compile time. This restriction was very severe. It meant that any loop headed, for example, by DO 10 I=1,N was not a candidate for automatic vectorization if N depended on the input data. The reason was that Cyber vector instructions allow for a maximum vector length of 65536, and the compiler did not generate code to check the size of N. The restriction could be overcome by specifying the "unsafe vectorization" option on the Fortran command, the effect of which was to shift responsibility to the programmer for ensuring that all vector lengths were within the limit. This undesirable and unnecessary burden on Fortran programmers was finally relieved in Cycle 661C of the compiler.

The Alliant, Convex and SCS (Cray) compilers accept directives in the form of special Fortran comment statements for aid in vectorizing a program. The pressure subdivision of BF3D contains 20 Cray compiler directives because this part of BF3D originated on a Cray 1. The directives were deactivated for the comparisons given in Table 1 because they are not applicable to other compilers. When activated, they increase the percentage of inner loops vectorized by the SCS (Cray) compiler to 89% for the pressure subdivision and 91% overall.

When the Alliant and Convex compilers meet an inner loop that they cannot vectorize fully, they attempt to rearrange code so that they can vectorize part of the loop. In Table 1 we counted a loop as vectorized if it was either fully or partially vectorized. The pressure subdivision was the only part of

BF3D that turned out to have partially vectorized loops. For the Alliant, the percentage of inner loops vectorized drops to 59% for the pressure subdivision and to 79% overall if only fully vectorized loops are counted. For the Convex, the percentages drop to 67% and 82%, respectively.

5. PERFORMANCE COMPARISONS

Detailed information for each benchmark run is given in the Appendix. Three similarly formatted tables are given for each computer, plus explanatory comments. In this section we draw some conclusions from the data.

Speed

The first table for each computer in the Appendix gives timing information in seconds for the pressure, evolution and particle subdivisions of BF3D; the sum of these, identified as PEP time; and total time for the execution of the benchmark, regarded as the sum of PEP time plus overhead within BF3D itself.

We see that the Cyber 205 is the fastest of the computers benchmarked. Table 2 exhibits the performance of the other computers relative to the Cyber. The performance ratios are obtained by dividing the PEP time for each computer by the PEP time for the Cyber. The Cyber, SCS, Convex and Alliant results are for 64-bit arithmetic; the Vax and Sun results are for 32-bit arithmetic.

	Small	Large	Giant	Dongarra
Cyber 205	1	1	1	1
SCS-40	1.3	1.4	1.6	2.1
Convex C1 XP	3.6	4.2	5.5	5.9
Alliant FX/8 (4 procs)	2.6	4.4	5.0	
Vax 11/785	25	36	41	44
Sun 3/160 (fpa)	34	49	67	29
Sun 3/160 (68881)	82	133	162	160

Table 2. Benchmark times relative to the Cyber 205.

Dongarra's ratios [8], normalized to the Cyber 205, are included for comparison. His benchmarks are for the problem of solving a dense linear system of order 100 by Gaussian elimination using Linpack [7] with Fortran versions of the Basic Linear Algebra subroutines. The precision of the arithmetic for the ratios cited is 32 bits for the Vax and Sun systems and 64 bits for the others. Dongarra does not give a benchmark result for a 4-processor Alliant system; for 8 processors and 1 processor the Dongarra ratio is 2.3 and 11, respectively.

Efficiency

The second table for each computer in the Appendix is derived from accounting data supplied by the operating systems: VSOS for the Cyber, CTSS for the SCS, VMS for the Vax, and implementations of Berkeley 4.2 Unix^ for the Convex, Alliant and Sun computers. We have tried to extract comparable information from these different systems:

User Time	The time used by the processor(s) in executing the benchmark program.
System Time	The time used by the processor(s) for system overhead such as page faulting. For the Vax, system time is not reported separately from user time.
Wallclock Time	The elapsed time from start of execution to end of execution.
Page Faults	The number of times a nonresident page had to be retrieved.
U+S to W	The ratio of user time plus system time to wallclock time.

We call the ratio of U+S to W the *job processing time ratio* (JPTR). The JPTR is not particularly revealing when other jobs coexist with the benchmark because it depends so strongly on characteristics of the workload. JPTRs for all the Convex and SCS benchmarks, the small and large Cyber

^UNIX is a registered trademark of AT&T Bell Laboratories.

benchmarks, and the small Vax benchmark fall into this category. When a job is run by itself, the JPTR becomes an approximate measure of the overall efficiency of the computer system for that job. JPTRs for all the Alliant and Sun benchmarks, the large and giant Vax benchmarks, and the giant Cyber benchmark fall into this category. In the Appendix, we identify the benchmarks that were run by themselves as *standalone benchmarks.*

For the standalone benchmarks, low JPTRs indicate bottlenecks in the system and/or network configuration. For example, not having a local disk on a Sun caused a noticeable drop in the efficiency for the giant benchmark: 55% for the Sun with floating-point accelerator and no local disk versus 79% for the Sun with local disk and the slower 68881 floating-point arithmetic, even though much more page faulting occurred on the latter system due to its smaller memory size.

Except for the SCS-40, which uses swapped memory management, all the systems use paged virtual memory management. The amount of I/O wait time associated with disk activity could have a significant impact on the JPTR, so a strong correlation between page faulting and the JPTR might be expected. However, we were unable to distinguish the number of page faults that actually resulted in disk reads or writes from the total number reported by the operating systems. In the case of the Sun benchmarks showing many page faults, the virtual size exceeded available physical memory and so disk activity was inevitable. In the case of the large Vax benchmark, enough physical memory was available so we suspect many of the page faults did not result in I/O activity. A similar conclusion is probably true for the giant Vax benchmark.

Effect of Vectorization

The third table for each computer in the Appendix shows a percentage breakdown of the PEP time into its three components, and also compares PEP time to total time. The particle code is quite insignificant compared to either the pressure or evolution codes. We observe that the pressure code is the most time-consuming, in general, on the vector computers, and as the problem size increases it becomes more dominant. However, on the scalar computers the situation is reversed. In fact, the evolution code dominates the pressure code for the scalar computers even more strongly than the opposite for the vector computers. We conclude that the evolution code is more effectively vectorized than the pressure code, resulting in a greater speedup. This is consistent with Table 1 which shows a higher percentage of the inner loops were vectorized in the evolution code regardless of compiler.

6. CONCLUSIONS

Four general conclusions are suggested by the results of our performance study.

First, the capability of compilers to vectorize Fortran loops automatically has achieved a high degree of success. It is interesting to observe that newer compilers (Alliant and Convex) outperform older compilers (Cyber and Cray) rather consistently. It follows from this observation that compiler directives will not be as important in the future as they have been in the past.

Second, the capability of compilers to parallelize code automatically shows promise for the future. The Alliant FX/8 multiplies the power of its vector processor, which is considerably slower than the Convex C1 XP processor, by effectively employing several of them to work simultaneously on a single task.

Third, Unix is becoming the standard operating system for supercomputers as well as scientific workstations and other types of computers. This is important because it provides a suitable basis on which to build advanced computing environments in which non-numerical processes can be integrated with very arithmetic-intensive processes. For example, 3D fluid calculations are typical of supercomputing applications in that appropriate graphical display becomes imperative for understanding. Graphics has been an afterthought, coming to mind only when the hopelessness of assimilating masses of printed numbers becomes apparent. Older operating systems, which configure the computing environment with the supercomputer accessible only in remote batch mode, will not be able to compete with newer, more powerful operating systems.

Fourth, the mini-supercomputers (SCS, Convex and Alliant) are impressive performers as compared to a representative mid-range mainframe (Vax), when utilized on problems that benefit from a vector architecture. They cannot match the processor speed of a full-fledged supercomputer (Cyber) but this is not

the only measure of utility. For example, if an individual research project could afford a dedicated mini-supercomputer then it would be free of competition for computer time from other projects. Runs could be scheduled according to the needs of the project rather than the priorities of a central computing facility.

We close with the observation that an extraordinary variety of computer hardware is available today compared with just a few years ago. The challenge, as always, is to select from it with wisdom and foresight.

7. ACKNOWLEDGMENTS

We wish to thank Alliant Computer Systems Corporation, Convex Computer Systems Corporation, and Scientific Computer Systems Corporation for generously providing time to run our benchmarks on their systems. In particular we wish to thank Ann McCue of SCS, Brian Christianson of Convex and John Dutton of Alliant. We wish also to thank Martin Knapp-Cordes of the Mathematical Analysis Division, NBS, for many long and useful discussions that contributed greatly to our understanding of computer operating systems.

References

1. H. R. Baum, R. G. Rehm, and G. W. Mulholland, "Computation of Fire Induced Flow and Smoke Coagulation," *19th International Symposium on Combustion*, pp. 921-931, The Combustion Institute, 1982.
2. H. R. Baum and R. G. Rehm, "Numerical Computation of Large-Scale Fire-Induced Flows," *Lecture Notes in Physics (E. Krause, Ed.)*, vol. 170, pp. 124-130, Springer-Verlag, New York, 1982.
3. H. R. Baum, R. G. Rehm, P. D. Barnett, and D. M. Corley, "Finite Difference Calculations of Buoyant Convection in an Enclosure, I: The Basic Algorithm," *SIAM J. Sci. Stat. Computing*, vol. 4, pp. 117-135, March 1983.
4. H. R. Baum and R. G. Rehm, "Finite Difference Solutions for Internal Waves in Enclosures," *SIAM J. Sci. Stat. Computing*, vol. 5, pp. 958-977, December 1984.
5. H. R. Baum, R. G. Rehm, D. M. Corley, and D. W. Lozier, "Time-Dependent Simulation of Small-Scale Turbulent Mixing and Reaction," *NBS Internal Report 86-3334*, National Bureau of Standards, Gaithersburg, MD 20899, November 1986.
6. H. R. Baum, D. M. Corley, and R. G. Rehm, "Time-Dependent Simulation of Small Scale Turbulent Mixing and Reaction," *21st International Symposium on Combustion*, The Combustion Institute, To appear.
7. J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, *LINPACK User's Guide*, SIAM Publications, Philadelphia, 1979.
8. J. J. Dongarra, "Performance of Various Computers Using Standard Linear Equations Software in a Fortran Environment," *Technical Memorandum 23*, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, November 10, 1986.
9. A. R. Karagozian and F. E. Marble, "Study of a Diffusion Flame in a Stretched Vortex," *Combustion Science and Technology*, vol. 45, pp. 65-84, 1986.
10. F. E. Marble, "Growth of a Diffusion Flame in the Field of a Vortex," *Recent Advances in Aerospace Sciences (C. Casci, Ed.)*, pp. 395-413, Plenum Publishing Corp., 1985.
11. R. G. Rehm and H. R. Baum, "The Equations of Motion for Thermally Driven, Buoyant Flows," *Journal of Research of the NBS*, vol. 83, pp. 297-308, May-June 1978.
12. R. G. Rehm, P. D. Barnett, H. R. Baum, and D. M. Corley, "Finite Difference Calculations of Buoyant Convection in an Enclosure: Verification of the Nonlinear Algorithm," *Applied Numerical Mathematics*, vol. 1, pp. 515 - 529, North-Holland, 1985.
13. R. G. Rehm, H. R. Baum, and D. W. Lozier, "Diffusion-Controlled Reaction in a Vortex Field," *NBS Internal Report 87-3572*, National Bureau of Standards, Gaithersburg, MD 20899, June 1987.

APPENDIX

The information given in this appendix is taken from the output produced by the benchmark programs and operating systems for each computer that we tested. Conclusions drawn from the data presented here are stated in Section 5 of this report.

Alliant System

An Alliant FX/8 computer system located at the office of Alliant Computer Systems Corporation in Littleton, Massachusetts, was used to execute the benchmark series in March 1987. The system had 192 megabytes of main memory, four vector processors, and 512 kilobytes of cache memory. The operating system for Alliant computers is Concentrix, an enhanced implementation of Berkeley 4.2 Unix. Fortran REAL arithmetic is normally 32-bit arithmetic on Alliant computers but the compiler has an option for specifying 64-bit REAL arithmetic without altering the source code. The data presented in the tables below are for 64-bit REAL arithmetic. The program was compiled with full optimization (-Ogvc) specified.

	Small	Large	Giant
Pressure (Pr)	19.6s	1902s	343s
Evolution (Ev)	11.2s	811s	143s
Particle (Pa)	3.9s	43s	1s
PEP (Pr+Ev+Pa)	34.7s	2756s	487s
Total (PEP+Overhead)	46.0s	2912s	494s

Table 3.1. Timing data from BF3D internal calls.

	Small	Large	Giant
User Time (U)	36.7s	2822s	490s
System Time (S)	9.6s	91s	5s
Wallclock Time (W)	56s	3098s	516s
Page Faults	26	71	26
U+S to W	83%^	94%^	96%^
^Standalone Benchmark.			

Table 3.2. Accounting data from the operating system.

	Small	Large	Giant
PEP to Total	75%	95%	99%
Pr to PEP	57%	69%	71%
Ev to PEP	32%	29%	29%
Pa to PEP	11%	2%	

Table 3.3. Breakdown by major BF3D component.

The data in Table 3.1 were obtained by calling the Unix timing function ETIME. The data in Table 3.2 were obtained by using the Unix command **time.** The data in Table 3.3 were derived from the data in Table 3.1.

We remark that the benchmark series was run also on a smaller FX/8 having 32 megabytes of main memory and 128 kilobytes of cache memory. The two systems were similar in other respects. We observed a slowdown in PEP time of 10% or less for each benchmark. This slowdown was due, in all likelihood, to the smaller cache memory, which probably restricted the flow of data to the processors.

Convex System

A Convex C1 XP computer system located at the office of Convex Computer Corporation in Greenbelt, Maryland, was used to execute the benchmark series in December 1986. The system had 32 megabytes of main memory and one vector processor. The operating system for Convex computers is Convex Unix, an enhanced version of Berkeley 4.2 Unix. Fortran REAL arithmetic is normally 32-bit arithmetic on the Convex but the compiler has an option for specifying 64-bit REAL arithmetic without altering the source code. The data presented in the tables below are for 64-bit REAL arithmetic. The program was compiled with full optimization (-O2) specified.

	Small	Large	Giant
Pressure (Pr)	20.6s	1556s	361s
Evolution (Ev)	17.4s	948s	170s
Particle (Pa)	8.9s	139s	2s
PEP (Pr+Ev+Pa)	46.9s	2643s	533s
Total (PEP+Overhead)	50.9s	2705s	542s

Table 4.1. Timing data from BF3D internal calls.

	Small	Large	Giant
User Time (U)	48.2s	2679s	522s
System Time (S)	2.8s	26s	21s
Wallclock Time (W)	103s	6177s	2152s
Page Faults	74	77	12
U+S to W	50%	44%	25%

Table 4.2. Accounting data from the operating system.

	Small	Large	Giant
PEP to Total	92%	98%	98%
Pr to PEP	44%	59%	68%
Ev to PEP	37%	36%	32%
Pa to PEP	19%	5%	

Table 4.3. Breakdown by major BF3D component.

The data in Table 4.1 were obtained by calling the Unix timing function ETIME. The data in Table 4.2 were obtained by using the Unix command **time.** The data in Table 4.3 were derived from the data in Table 4.1.

We remark that the benchmark series was run also in 32-bit REAL arithmetic, for which we observed an approximate 25% reduction in PEP time. Also, the benchmarks were rerun in May 1987 with a new version of the Fortran compiler. Nearly a 10% speedup was observed for the large benchmark, and nearly 5% for the small and giant benchmarks.

Cyber System

The CDC Cyber 200 Model 205 Series 600 computer system at the National Bureau of Standards in Gaithersburg, Maryland, has 4 million 64-bit words of main memory (32 megabytes) and 2 vector pipelines for floating-point arithmetic (Cyber 205 systems have only one processor; the effect of the second pipeline is to double the pipeline speed). The computer runs in batch mode under VSOS (Virtual Storage Operating System). Jobs are submitted via NOS (Network Operating System) from a Cyber 180/855. The benchmark series was run in February 1987 to obtain the data given in the tables below. Fortran REAL arithmetic, i.e. 64-bit arithmetic on the Cyber, was used. The program was compiled using full optimization (OPT=DPRSV).

	Small	Large	Giant
Pressure (Pr)	5.4s	358s	57.1s
Evolution (Ev)	5.9s	247s	40.3s
Particle (Pa)	1.8s	18s	0.2s
PEP (Pr+Ev+Pa)	13.1s	623s	97.6s
Total (PEP+Overhead)	13.6s	628s	99.0s

Table 5.1. Timing data from BF3D internal calls.

	Small	Large	Giant
User Time (U)	15.2s	630s	101s
System Time (S)	2.1s	2s	2s
Wallclock Time (W)	119s	1458s	107s
Page Faults	16	39	53
U+S to W	15%	43%	96%*
*Standalone Benchmark.			

Table 5.2. Accounting data from the operating system.

	Small	Large	Giant
PEP to Total	96%	99%	99%
Pr to PEP	41%	57%	59%
Ev to PEP	45%	40%	41%
Pa to PEP	14%	3%	

Table 5.3. Breakdown by major BF3D component.

The data in Table 5.1 were obtained by calling the system timing function SECOND. The data in Table 5.2 were obtained by using the VSOS command SUMMARY; the number of virtual system requests is identified here as the number of page faults. The data in Table 5.3 were derived from the data in Table 5.1.

SCS System

An SCS-40 computer system in San Diego was used to execute the benchmark series in March 1987. The system had 32 megabytes of main memory and one vector processor. The system was run as a batch processor with a Vax for terminal access. The operating system was CTSS (Cray Time Sharing System). The company is developing a version of Unix as well as COS (Cray Operating System) for their computers, and also is developing the SCS-40 to obviate the need for a frontend. The SCS-40 is designed to be highly compatible with the Cray X-MP architecture, in fact sharing the same instruction set and Fortran compiler. The data presented in the tables below are for 64-bit REAL arithmetic.

	Small	Large	Giant
Pressure (Pr)	7.4s	439s	87s
Evolution (Ev)	7.3s	372s	65s
Particle (Pa)	2.5s	39s	
PEP (Pr+Ev+Pa)	17.2s	850s	152s
Total (PEP+Overhead)	19.6s	905s	156s

Table 6.1. Timing data from BF3D internal calls.

	Small	Large	Giant
User Time (U)	19.7s	903s	157s
System Time (S)	0.1s	3s	
Wallclock Time (W)	77s	2358s	392s
U+S to W	26%	38%	40%

Table 6.2. Accounting data from the operating system.

	Small	Large	Giant
PEP to Total	88%	94%	97%
Pr to PEP	43%	51%	57%
Ev to PEP	42%	44%	43%
Pa to PEP	15%	5%	

Table 6.3. Breakdown by major BF3D component.

The data in Table 6.1 were obtained by calling the system timing function SECOND. The data in Table 6.2 were obtained from the CTSS accounting log. The data in Table 6.3 were derived from the data in Table 6.1.

Sun System With FPA

The benchmark series was run in March 1987 on a diskless Sun 3/160 computer at the National Bureau of Standards. The fileserver was a Sun 3/180S with Fujitsu Eagle disk, connected via an Ethernet local area network. The system had 8 megabytes of main memory and a floating-point accelerator (fpa). The fpa uses a Weitek floating-point chip set for enhanced performance. The operating system was Sun Unix 4.2, release 3.1, an enhanced version of Berkeley 4.2 Unix. Fortran REAL arithmetic uses a 32-bit wordlength, and the data presented in the tables below are for 32-bit arithmetic. The program was compiled with full optimization (-O) specified.

	Small	Large	Giant
Pressure (Pr)	94s	7753s	1399s
Evolution (Ev)	314s	22259s	5149s
Particle (Pa)	36s	659s	26s
PEP (Pr+Ev+Pa)	444s	30671s	6574s
Total (PEP+Overhead)	461s	31320s	6651s

Table 7.1. Timing data from BF3D internal calls.

	Small	Large	Giant
User Time (U)	438s	29818s	5331s
System Time (S)	24s	1503s	1320s
Wallclock Time (W)	488s	33955s	12086s
Page Faults	0	259	81334
U+S to W	95%^	92%^	55%^
^Standalone Benchmark.			

Table 7.2. Accounting data from the operating system.

	Small	Large	Giant
PEP to Total	96%	98%	99%
Pr to PEP	21%	25%	22%
Ev to PEP	71%	73%	78%
Pa to PEP	8%	2%	

Table 7.3. Breakdown by major BF3D component.

The data in Table 7.1 were obtained by calling the Unix timing function ETIME. The data in Table 7.2 were obtained by using the Unix command **time.** The data in Table 7.3 were derived from the data in Table 7.1.

Sun System With 68881

The benchmark series was run in March 1987 on a Sun 3/160 computer at the National Bureau of Standards. The system had 4 megabytes of main memory, a Motorola 68881 floating-point chip, and a local Fujitsu Eagle disk. The operating system was Sun Unix 4.2, release 3.1, an enhanced version of Berkeley 4.2 Unix. Fortran REAL arithmetic uses a 32-bit wordlength, and the data presented in the tables below are for 32-bit arithmetic. The program was compiled with full optimization (-O) specified.

	Small	Large	Giant
Pressure (Pr)	281s	23172s	4207s
Evolution (Ev)	695s	57966s	11513s
Particle (Pa)	100s	2002s	45s
PEP (Pr+Ev+Pa)	1076s	83140s	15765s
Total (PEP+Overhead)	1094s	83900s	15935s

Table 8.1. Timing data from BF3D internal calls.

	Small	Large	Giant
User Time (U)	1090s	79733s	14293s
System Time (S)	5s	4167s	1642s
Wallclock Time (W)	1136s	101361s	20148s
Page Faults	0	645061	157112
U+S to W	96%^	83%^	79%^
^Standalone Benchmark.			

Table 8.2. Accounting data from the operating system.

	Small	Large	Giant
PEP to Total	98%	99%	99%
Pr to PEP	26%	28%	27%
Ev to PEP	65%	70%	73%
Pa to PEP	9%	2%	

Table 8.3. Breakdown by major BF3D component.

The data in Table 8.1 were obtained by calling the Unix timing function ETIME. The data in Table 8.2 were obtained by using the Unix command **time.** The data in Table 8.3 were derived from the data in Table 8.1.

Vax System

A Vax 11/785 computer system with 16 megabytes of main memory, running under the VMS operating system at the National Bureau of Standards, was used to execute the benchmark series in April 1987. Fortran REAL arithmetic is executed in 32-bit words on the Vax, and the results given in the tables below are for 32-bit arithmetic. The program was compiled with full optimization, i.e. the default option for the Vax Fortran compiler.

	Small	Large	Giant
Pressure (Pr)	112s	9133s	1672s
Evolution (Ev)	178s	12731s	2365s
Particle (Pa)	40s	719s	7s
PEP (Pr+Ev+Pa)	330s	22583s	4044s
Total (PEP+Overhead)	342s	22770s	4087s

Table 9.1. Timing data from BF3D internal calls.

	Small	Large	Giant
User+System Time (U+S)	346s	22778s	4096s
Wallclock Time (W)	514s	24659s	4347s
Page Faults	2126	21724	72330
U+S to W	67%	92%^	94%^
^Standalone Benchmark.			

Table 9.2. Accounting data from the operating system.

	Small	Large	Giant
PEP to Total	96%	99%	99%
Pr to PEP	34%	40%	41%
Ev to PEP	54%	57%	59%
Pa to PEP	12%	3%	

Table 9.3. Breakdown by major BF3D component.

The data in Table 9.1 were obtained by calling the VMS timing function LIB$STAT_TIMER. The data in Table 9.2 were obtained from the VMS log file. The data in Table 9.3 were derived from the data in Table 9.1. The operating system was configured so as to allow working sets of up to 10.24 megabytes, which is nearly enough for the giant benchmark to reside fully in memory.

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBSIR 87-3638	2. Performing Organ. Report No.	3. Publication Date September 1987

4. TITLE AND SUBTITLE

Some Performance Comparisons for a Fluid Dynamics Code

5. AUTHOR(S)

D. W. Lozier and R. G. Rehm

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

The 3D transient motion of a buoyancy-driven perfect gas in an enclosure is computed by a Fortran program (BF3D). A combustion model for eventual inclusion in this program is under development. BF3D changes slowly, has a long lifetime, and is run fairly infrequently. Typical runs have large storage and moderate CPU requirements. BF3D runs on large supercomputers but the newer mini-supercomputers appear to be suitable also and may be advantageous for ease of access and usage. Scientific workstations are convenient for development. Comparisons of BF3D on selected supercomputers, mini-supercomputers, scientific workstations and conventional mainframes are presented for the purpose of making partial benchmark data available to the computing public.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

computer performance comparisons; mini-supercomputer performance; scientific workstation performance; supercomputer performance; three-dimensional CFD benchmark program; vectorizing compiler performance

13. AVAILABILITY

[X] Unlimited
[ ] For Official Distribution. Do Not Release to NTIS
[ ] Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.
[X] Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

19

15. Price

$9.95

USCOMM-DC 6043-P80